

12/30/98  
jc614 U.S. PTO

Attorney Docket No.: MIPS-458.CON.

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**  
**Patent Application**

I hereby certify that this transmittal of the below described documents is being deposited with the United States Postal Service in an envelope bearing Express Mail Postage and an Express Mail label, with the below serial number, addressed to the Assistant Commissioner of Patents, Washington, D.C., 20231, on the below date of deposit.

Express Mail Label No.:	EE795865202US	Name of Person Making the Deposit:	TONY CHOU
Date of Deposit:	12/30/98	Signature of the Person Making the Deposit:	<i>Tony Chou</i>

jc596 U.S. PTO  
09/22/3046  
12/30/98

Anticipated Classification of this application:

Class \_\_\_\_\_ Subclass \_\_\_\_\_

Prior application: \_\_\_\_\_

Examiner: Shah, A.

Art Unit: 2783

Box Patent Application  
Assistant Commissioner for Patents  
Washington, D.C. 20231

**TRANSMITTAL OF FILING UNDER 37 CFR 1.53(b)**

This is a request for filing a Continuation application under 37 CFR 1.53(b), of pending prior application

Serial No. 08/947,648 filed on 10/09/97  
of Timothy J. van Hook, Peter Hsu, William A. Huffman, Henry P. Moreton  
and Earl A. Killian  
*Inventor(s)*  
for A METHOD FOR PROVIDING EXTENDED PRECISION IN SIMD VECTOR ARITHMETIC  
OPERATIONS  
*Title of invention*

**Copy of Prior Application as Filed That is Attached**

☒ A copy of the above identified prior application, including the oath or declaration originally filed, with no new matter added is attached.

The copy of the papers of prior application as filed which are attached are as follows:

<input checked="" type="checkbox"/>	22	page(s) of specification
<input checked="" type="checkbox"/>	7	page(s) of claims
<input checked="" type="checkbox"/>	1	page(s) of abstract
<input checked="" type="checkbox"/>	9	sheet(s) of drawing
<input checked="" type="checkbox"/>	15	page(s) of declaration and power of attorney

☒ in accordance with the indication required by 37 CFR 1.53(b), my records reflect that the original signed declaration showing applicant's signature was filed on 05/11/98, 03/04/98, 03/05/98, 03/06/98, 03/20/98 \_\_\_\_\_

☐ the amendment referred to in the declaration filed to complete the prior application and I hereby state that this amendment did not introduce new matter therein.

### Amendments

- ☒ Cancel in this application original claims ..... 2-40 ..... of the prior application before calculating the filing fee. (At least one original independent claim must be retained for filing purposes)
- ☐ A preliminary Amendment is enclosed. (Claims added by this amendment have been properly numbered consecutively beginning with the number next following the highest numbered original claim in the prior application)

### Information Disclosure Statement

- ☐ An information disclosure statement is submitted herewith

### Drawings

- ☒ Drawings are enclosed
- ☐ formal
- ☒ informal

### Priority Claim

#### 35 U.S.C. 119

- ☐ Priority of application Serial Number ..... filed on ..... in ..... is claimed under 35 U.S.C. 119.
- ☐ The certified copy has been filed in prior U.S. application Serial No. .... in .....
- ☐ The certified copy will follow. ....

#### 35 U.S.C. 120, 121 and 365(c)

"This application is a continuation of and claims the benefit of copending application(s)

- ☒ application number 08/947,648 filed on 10/09/97
- ☐ International Application ..... filed on .....

and which designated in the U.S."

### Relate Back

- ☒ Amend this specification by inserting, before the first line, the following sentence:

"This is a continuation of copending application(s)

- ☒ Serial Number 08/947,648 filed on 10/09/97

which is hereby incorporated by reference to this specification

- ☐ International Application ..... filed on .....

which designated the U.S."

### Inventorship Statement

(a) With respect to prior copending U.S. application from which this application claims benefit under 35 USC 120 the inventor(s) in this application are:

- ☒ the same.  
☐ less than those named in the prior application. It is requested that the following inventor(s) identified above for the prior application be deleted:

.....  
*Type names of inventors to be deleted*

(b) The inventorship for all the claims in this application are

- ☒ the same.  
☐ not the same. And an explanation, including the ownership of the various claims at the time the last claimed invention was made, is submitted.

### Assignment

- ☒ The prior application is assigned of record to

- ☐ An assignment of the invention to

.....  
is attached. A separate "Cover Sheet for Assignment (Document) Accompanying New Patent Application" or Form PTO 1595 is also Attached.

### Power of Attorney

- ☒ The power of attorney in the prior application is to

James P. Hao	36,398
Attorney	Reg. No.
Anthony C. Murabito	35,295
Attorney	Reg. No.
John P. Wagner, Jr.	35,398
Attorney	Reg. No.
Glenn D. Barnes	42,293
Attorney	Reg. No.
Wilfred H. Lam	41,923
Attorney	Reg. No.
Patrick W. Ma	P-44,215
Attorney	Reg. No.

- a. ☒ The power appears in the original papers in the prior application.  
b. ☐ Because the power does not appear in the original papers, a copy of the power in the prior application is enclosed.  
c. ☐ A new power has been executed and is attached.  
d. ☐ Address all future communications to

**Maintenance of Codependency of Prior Application**

- ☐ A petition, fee and response has been filed to extend the term pending prior application until \_\_\_\_\_  
☐ A copy of the petition for extension of time in the prior application is attached.

**Conditional Petition for Extension of Time in Prior Application**

- ☒ A conditional petition for extension of time is being filed in the pending parent application.  
☐ A copy of the conditional petition for extension of time in the prior application is attached

**Abandonment of Prior Application (if applicable)**

- ☐ Please abandon the prior application at a time while the prior application is pending or when the petition for extension of time or to revive in that application is granted and when this application is granted a filing date so as to make this application copending with said prior application.

**Notification in Parent Application of the Filing of This Continuation Application**

- ☒ A notification of the filing of this continuation is being filed in the parent application from which this application claims priority under 35 USC § 120.

**Statement by Assignee (if applicable)**

- ☐ In accordance with 37 CFR 3.73, I have reviewed the evidentiary documents establishing my/our ownership of the application identified herein, and certify that to the best of my/our knowledge and belief, title is with me/us who seek to take action.  
☐ Assignment submitted herewith for recordal

I hereby declare further that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Type or print name of person signing declaration .....

Signature ..... Date .....

Residence ..... Citizenship .....  
(City) (State)

P.O. Address .....

.....  
(type name of assignee)

.....  
Title of person authorized to sign on behalf of assignee

.....  
Address of Assignee

Assignment recorded in PTO on ..... Reel ..... Frame .....

The statement under 37 CFR 3.73(b)

- ☐ has been filed in the parent application.  
☐ a copy of the statement previously filed in the parent application is attached.

## Fee Calculation

CLAIMS AS FILED					
	NO. OF CLAIMS		EXTRA CLAIMS	RATE	FEES
Basic Application Fee					\$760.00
Total Claims	1	Minus 40=	0	X \$18 =	\$0.00
Independent Claims	1	Minus 3=	0	X \$78 =	\$0.00
If multiple dependent claims are presented, add \$260.00					\$0.00
<b>TOTAL APPLICATION FEE DUE</b>					<b>\$760.00</b>

## PAYMENT OF FEES

1. The full fee due in connection with this communication is \_ provided as follows:

Not enclosed

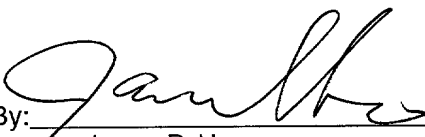
☒ [ X ] No filing fee is to be paid at this time.

Please direct all correspondence concerning the above-identified application to the following address:

**WAGNER, MURABITO & HAO**  
Two North Market Street, Third Floor  
San Jose, California 95113  
(408) 938-9060

Respectfully submitted,

Date: 12/20/98

By:   
James P. Hao  
Reg. No. 36,398



## IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

I hereby certify that this transmittal of the below described document is being deposited with the United States Postal Service in an envelope bearing First Class Postage and addressed to the Assistant Commissioner for Patents, Washington, D.C., 20231, on the below date of deposit.				
Date of Deposit:	12/30/98	Name of Person Making the Deposit:	KATHERINE RINALDI	Signature of the Person Making the Deposit: <i>Katherine Rinaldi</i>

In re Application

Inventor(s): Timothy J. van Hook, Peter Hsu, William A. Huffman, Henry P. Moreton and Earl A. Killian

Application No.: 08/947,648

Filed: 10/09/97

Title: A METHOD FOR PROVIDING EXTENDED PRECISION IN SIMD VECTOR ARITHMETIC OPERATIONS

Assistant Commissioner for Patents  
Washington, D.C. 20231

## NOTIFICATION OF FILING OF CONTINUING APPLICATION

☒ Notification is hereby being made of the filing of a continuation application for this case☒ concurrently herewith.  
☐ on .....

Please direct all correspondence concerning the above-identified application to the following address:

**WAGNER, MURABITO & HAO**  
Two North Market Street, Third Floor  
San Jose, California 95113  
(408) 938-9060

Respectfully submitted,

Date: \_\_\_\_\_

*12/30/98*

By: \_\_\_\_\_

*James P. Hao*James P. Hao  
Reg. No. 36,398

860831-5462250

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

I hereby certify that this transmittal of the below described document is being deposited with the United States Postal Service in an envelope bearing First Class Postage and addressed to the Assistant Commissioner for Patents, Washington, D.C., 20231, on the below date of deposit.			
Date of Deposit:	12/30/98	Name of Person Making the Deposit:	KATHERINE RINALDI
		Signature of the Person Making the Deposit:	<i>Katherine Rinaldi</i>

In re Application of:

Inventor(s): Timothy J. van Hook, Peter Hsu, William A. Huffman, Henry P. Moreton and Earl A. Killian

Application No.: 08/947,648

Filed: 10/09/97

Title: A METHOD FOR PROVIDING EXTENDED PRECISION IN SIMD VECTOR ARITHMETIC OPERATIONS

Assistant Commissioner for Patents  
Washington, D.C. 20231

CONDITIONAL PETITION FOR EXTENSION OF TIME

This conditional petition is being filed concurrently with the filing of a CONTINUATION APPLICATION

and provides for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

**Conditional petition for extension of time**

If any extension of time for the accompanying response is required, applicant requests that this be considered a petition therefor.

**Status**

This application is on behalf of

- ☒ other than a small entity  
☐ a small entity

A verified statement:

- ☐ is attached  
☐ is already filed

Please direct all correspondence concerning the above-identified application to the following address:

**WAGNER, MURABITO & HAO**  
Two North Market Street, Third Floor  
San Jose, California 95113  
(408) 938-9060

Respectfully submitted,

Date: 12/30/98

By: *James P. Hao*  
James P. Hao  
Reg. No. 36,398



UNITED STATES PATENT APPLICATION  
FOR

A METHOD FOR PROVIDING EXTENDED PRECISION IN SIMD VECTOR  
ARITHMETIC OPERATIONS

Inventors:

Timothy van Hook

Peter Hsu

William Huffman

Henry Moreton

Earl Killian

Prepared by:

WAGNER, MURABITO & HAO

Two North Market Street

Third Floor

San Jose, California 95113

# A METHOD FOR PROVIDING EXTENDED PRECISION IN SIMD VECTOR ARITHMETIC OPERATIONS

## FIELD OF THE INVENTION

5           The present claimed invention relates to the field of single instruction multiple data (SIMD) vector process. More particularly, the present claimed invention relates to extended precision in SIMD vector arithmetic operations.

## BACKGROUND ART

10           Today, most processors in computer systems provide a 64-bit datapath architecture. The 64-bit datapath allows operations such as read, write, add, subtract, and multiply on the entire 64 bits of data at a time. This added bandwidth has significantly improved performance of the processors.

15           However, the data types of many real world applications do not utilize the full 64 bits in data processing. For example, in digital signal processing (DSP) applications involving audio, video, and graphics data processing, the light and sound values are usually represented by data types of 8, 12, 16, or 24 bit numbers. This is because people typically are not able to distinguish the  
20 levels of light and sound beyond the levels represented by these numbers of bits. Hence, DSP applications typically require data types far less than the full 64 bits provided in the datapath in most computer systems.

          In initial applications, the entire datapath was used to compute an  
25 image or sound values. For example, an 8 or 16 bit number representing a pixel or sound value was loaded into a 64-bit number. Then, an arithmetic operation, such as an add or multiply, was performed on the entire 64-bit

number. This method proved inefficient however, as it was soon realized that not all the data bits were being utilized in the process since digital representation of a sound or pixel requires far fewer bits. Thus, in order to utilize the entire datapath, a multitude of smaller numbers were packed into  
5 the 64 bit doubleword.

Furthermore, much of data processing in DSP applications involve repetitive and parallel processing of small integer data types using loops. To take advantage of this repetitive and parallel data process, a number of today's  
10 processors implements single instruction multiple data (SIMD) in the instruction architecture. For instance, the Intel Pentium MMX<sup>TM</sup> chips incorporate a set of SIMD instructions to boost multimedia performance.

Prior Art Figure 1 illustrates an exemplary single instruction multiple  
15 data instruction process. Exemplary registers, vs and vt, in a processor are of 64-bit width. Each register is packed with four 16-bit data elements fetched from memory: register vs contains vs[0], vs[1], vs[2], and vs[3] and register vt contains vt[0], vt[1], vt[2], and vt[3]. The registers in essence contain a vector of N elements. To add elements of matching index, an add instruction adds,  
20 independently, each of the element pairs of matching index from vs and vt. A third register, vd, of 64-bit width may be used to store the result. For example, vs[0] is added to vt[0] and its result is stored into vd[0]. Similarly, vd[1], vd[2], and vd[3] store the sum of vs and vd elements of corresponding indexes. Hence, a single add operation on the 64-bit vector results in 4  
25 simultaneous additions on each of the 16-bit elements. On the other hand, if 8-bit elements were packed into the registers, one add operation performs 8 independent additions in parallel. Consequently, when a SIMD arithmetic

instruction, such as addition, subtraction, or multiply, is performed on the data in the 64-bit datapath, the operation actually performs multiple numbers of operations independently and in parallel on each of the smaller elements comprising the 64 bit datapath.

5

Unfortunately however, an arithmetic operation such as add and multiply on SIMD vectors typically increases the number of significant bits in the result. For instance, an addition of two  $n$ -bit numbers may result in a number of  $n+1$  bits. Moreover, a multiplication of two  $n$ -bit numbers  
10 produces a number of  $2n$  bit width. Hence, the results of an arithmetic operation on a SIMD vector may not be accurate to a desired significant bit.

Furthermore, the nature of multimedia DSP applications often increases inaccuracies in significant bits. For example, many DSP algorithms  
15 implemented in DSP applications require a series of computations producing partial results that are larger or bigger, in terms of significant number of bits, than the final result. Since the final result does not fully account for the significant bits of these partial results, the final result may not accurately reflect the ideal result, which takes into account all significant bits of the  
20 intermediate results.

To recapture the full significant bits in a SIMD vector arithmetic operation, the size of the data in bits for each individual element was typically boosted or promoted to twice the size of the original data in bits. Thus, for  
25 multiplication on 8-bit elements in a SIMD vector for instance, the 8-bit elements were converted (i.e., unpacked) into 16-bit elements containing 8 significant bits to provide enough space to hold the subsequent product.

Unfortunately however, the boost in the number of data bits largely undermined the benefits of SIMD vector scheme by reducing the speed of an arithmetic operation in half. This is because the boosting of data bits to twice  
5 the original size results in half as many data elements in a register. Hence, an operation on the entire 64-bit datapath comprised of 16-bit elements accomplishes only 4 operations in comparison to 8 operations on a 64-bit datapath comprised of 8-bit elements. In short, boosting a data size by X-fold results in performance reduction of  $(1/X)*100$  percent. As a result, instead of  
10 an effective 64-bit datapath, the effective datapath was only 32-bits wide.

Thus, what is needed is a method and system for providing extended precision in SIMD vector arithmetic operations without sacrificing speed and performance.  
15

## SUMMARY OF THE INVENTION

5 The present invention provides extended precision in SIMD arithmetic operations in a processor having a register file and an accumulator. The register file is comprised of a plurality of general purpose registers of N bit width. The size of the accumulator is preferably an integer multiple of the size of the general purpose registers. The preferred embodiment uses registers of 64 bits and an accumulator of 192 bits. The present invention first loads, from a memory, a first set of data elements into a first vector register and a second set of data elements into a second vector register. Each data element comprises N bits. Next, an arithmetic instruction is fetched from memory and is decoded. Then, a first vector register and a second vector register are read from the register file as specified in the arithmetic instruction. The present invention then executes the arithmetic instruction on corresponding data elements in the first and second vector registers. The result of the execution is then written into the accumulator. Then, each element in the accumulator is transformed into an N-bit width element and written into a third register for further operation or storage in the memory.

20 In an alternative embodiment, the accumulator contains a third set of data elements. After the arithmetic operation between the data elements in the first and second vector registers, the result of the execution is added to the corresponding elements in the accumulator. The result of the addition is then written into the accumulator. Thereafter, each element in the accumulator is transformed into an N-bit width element and written into a third register for further operation or storage in the memory.

## BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

5

Prior Art Figure 1 illustrates an exemplary single instruction multiple data (SIMD) instruction method.

Figure 2 illustrates an exemplary computer system of the present invention.

10 Figure 3 illustrates a block diagram of an exemplary datapath including a SIMD vector unit (VU), a register file, and a vector load/store unit according to one embodiment of the present invention.

Figure 4 illustrates a more detailed datapath architecture including the accumulator in accordance with the present invention.

15 Figure 5 illustrates a flow diagram of general operation of an exemplary arithmetic instruction according to a preferred embodiment of the present invention.

Figure 6 illustrates element select format for 4 16-bit elements in a 64-bit register.

20 Figure 7 illustrates element select format for 8 8-bit elements in a 64-bit register.

Figure 8 illustrates an exemplary ADDA.fmt arithmetic operation between elements of exemplary operand registers vs and vt.

25 Figure 9 illustrates an exemplary ADDL.fmt arithmetic operation between elements of exemplary operand registers vs and vt.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details. In other instances well known methods, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

The present invention features a method for providing extended precision in single-instruction multiple-data (SIMD) arithmetic operations in a computer system. The preferred embodiment of the present invention performs integer SIMD vector arithmetic operations in a processor having 64-bit wide datapath within an exemplary computer system described above. Extended precision in the SIMD arithmetic operations are supplied through the use of an accumulator register having a preferred width of 3 times the general purpose register width. Although a datapath of 64-bits is exemplified herein, the present invention is readily adaptable to datapaths of other variations in width.

## COMPUTER SYSTEM ENVIRONMENT

Figure 2 illustrates an exemplary computer system 212 comprised of a system bus 200 for communicating information, one or more central processors 201 coupled with the bus 200 for processing information and instructions, a computer readable volatile memory unit 202 (e.g., random access memory, static RAM, dynamic RAM, etc.) coupled with the bus 200 for storing information and instructions for the central processor(s) 201, a



computer readable non-volatile memory unit (e.g., read only memory, programmable ROM, flash memory, EPROM, EEPROM, etc.) coupled with the bus 200 for storing static information and instructions for the processor(s).

5 Computer system 212 of Figure 2 also includes a mass storage computer readable data storage device 204 (hard drive, floppy, CD-ROM, optical drive, etc.) such as a magnetic or optical disk and disk drive coupled with the bus 200 for storing information and instructions. Optionally, system 212 can include a display device 205 coupled to the bus 200 for displaying information to the  
10 user, an alphanumeric input device 206 including alphanumeric and function keys coupled to the bus 200 for communicating information and command selections to the central processor(s) 201, a cursor control device 207 coupled to the bus for communicating user input information and command selections to the central processor(s) 201, and a signal generating device 208  
15 coupled to the bus 200 for communicating command selections to the processor(s) 201.

According to a preferred embodiment of the present invention, the processor(s) 201 is a SIMD vector unit which can function as a coprocessor for  
20 a host processor (not shown). The VU performs arithmetic and logical operations on individual data elements within a data word using the instruction methods described below. Data words are treated as vectors of  $N \times 1$  elements, where  $N$  can be 8, 16, 32, 64, or multiples thereof. For example, a set of  $N \times 1$  data elements of either 8- or 16-bit fields comprises a data  
25 doubleword of 64-bit width. Hence, a 64 bit wide double word contains either 4 16-bit elements or 8 8-bit elements.

Figure 3 illustrates a block diagram of an exemplary datapath 300 including a SIMD vector unit (VU) 302, a register file 304, a vector load/store unit 318, and crossbar circuits 314 and 316 according to one embodiment of the present invention. The VU 302 executes an operation specified in the instruction on each element within a vector in parallel. The VU 302 can operate on data that is the full width of the local on-chip memories, up to 64 bits. This allows parallel operations on 8 8-bit, 4 16-bit, 2 32-bit, or 1 64-bit elements in one cycle. The VU 302 includes an accumulator 312 to hold values to be accumulated or accumulated results.

The vector register file is comprised of 32 64-bit general purpose registers 306 through 310. The general purpose registers 306 through 310 are visible to the programmer and can be used to store intermediate results. The preferred embodiment of the present invention uses the floating point registers (FGR) of a floating point unit (FPU) as its vector registers.

In this shared arrangement, data is moved between the vector register file 304 and memory with Floating Point load and store doubleword instructions through the vector load/store unit 318. These load and store operations are unformatted. That is, no format conversions are performed and therefore no floating-point exceptions can occur due to these operations. Similarly, data is moved between the vector register file 304 and the VU 302 without format conversions, and thus no floating-point exception occurs.

Within each register, data may be written, or read, as bytes (8-bits), short-words (16-bits), words (32-bits), or double-words (64-bits). Specifically, the vector registers of the present invention are interpreted in the following

new data formats: Quad Half (QH), Oct Byte (OB), Bi Word (BW), and Long (L). In QH format, a vector register is interpreted as having 16-bit elements. For example, a 64-bit vector register is interpreted as a vector of 4 signed 16-bit integers. OB format interprets a vector register as being comprised of 8-bit elements. Hence, an exemplary 64-bit vector register is seen as a vector of 8 unsigned 8-bit integers. In BW format, a vector register is interpreted as having 32-bit elements. L format interprets a vector register as having 64-bit elements. These data types are provided to be adaptable to various register sizes of a processor. As described above, data format conversion is not necessary between these formats and floating-point format.

With reference to Figure 3, the present invention utilizes crossbar circuits to select and route elements of a vector operand. For example, the crossbar circuit 314 allows selection of elements of a given data type and pass on the selected elements as operands to VU 302. The VU 302 performs arithmetic operations on operands comprised of elements and outputs the result to another crossbar circuit 316. This crossbar circuit 316 routes the resulting elements to corresponding element fields in registers such as vd 310 and accumulator 312. Those skilled in the art will no doubt recognize that crossbar circuits are routinely used to select and route the elements of a vector operand.

With reference to Figure 3, the present invention also provides a special register, accumulator 312, of preferably 192-bit width. This register is used to store intermediate add, subtract, or multiply results generated by one instruction with the intermediate add, subtract, or multiply results generated by either previous or subsequent instructions. The accumulator 312 can also

be loaded with a vector of elements from memory through a register. In addition, the accumulator 312 is capable for forwarding data to the VU 302, which executes arithmetic instructions. Although the accumulator 312 is shown to be included in the VU 302, those skilled in the art will recognize that it can also be placed in other parts of the datapath so as to hold either accumulated results or values to be accumulated.

Figure 4 illustrates a more detailed datapath architecture including the accumulator 312. In this datapath, the contents of two registers, vs and vt, are operated on by an ALU 402 to produce a result. The result from the ALU can be supplied as an operand to another ALU such as an adder/subtractor 404. In this datapath configuration, the accumulator 312 can forward its content to be used as the other operand to the adder/subtractor 404. In this manner, the accumulator 312 can be used as both a source and a destination in consecutive cycles without causing pipe stalls or data hazards. By thus accumulating the intermediate results in its expanded form in tandem with its ability to be used as both a source and a destination, the accumulator 312 is used to provide extended precision for SIMD arithmetic operations.

An exemplary accumulator of the present invention is larger in size than general purpose registers. The preferred embodiment uses 192-bit accumulator and 64-bit registers. The format of the accumulator is determined by the format of the elements accumulated. That is, the data types of an accumulator matches the data type of operands specified in an instruction. For example, if the operand register is in QH format, the accumulator is interpreted to contain 4 48-bit elements. In OB format, the accumulator is seen as having 8 24-bit elements. In addition, accumulator

elements are always signed. Elements are stored from or loaded into the accumulator indirectly to and from the main memory by staging the elements through the shared Floating Point register file.

5        Figure 5 illustrates a flow diagram of an exemplary arithmetic operation according to a preferred embodiment of the invention. In step 502, an arithmetic instruction is fetched from memory into an instruction register. Then in step 504, the instruction is decoded to determine the specific arithmetic operation, operand registers, selection of elements in operand  
10    registers, and data types. The instruction opcode specifies an arithmetic operation such as add, multiply, or subtract in its opcode field. The instruction also specifies the data type of elements, which determines the width in bits and number of elements involved in the arithmetic operation. For example, OB data type format instructs the processor to interpret a vector  
15    register as containing 8 8-bit elements. On the other hand, QH format directs the processor to interpret the vector register as having 4 16-bit elements.

      The instruction further specifies two operand registers, a first register (vs) and a second register (vt). The instruction selects the elements of the  
20    second register, vt, to be used with each element of the accumulator, and/or the first register, vs. For example, the present invention allows selection of one element from the second register to be used in an arithmetic operation with all the elements in the first register independently and in parallel. The selected element is replicated for every element in the first register. In the  
25    alternative, the present invention provides selection of all elements from the second register to be used in the arithmetic operation with all the elements in the first register. The arithmetic operation operates on the corresponding

elements of the registers independently and in parallel. The present invention also provides an immediate value (i.e., a constant) in a vector field in the instruction. The immediate value is replicated for every element of the second register before an arithmetic operation is performed between the first and second registers.

According to the decoded instruction, the first register and the second register with the selected elements are read for execution of the arithmetic operation in step 506. Then in step 508, the arithmetic operation encoded in the instruction is executed using each pair of the corresponding elements of first register and the second register as operands. The resulting elements of the execution are written into corresponding elements in the accumulator in step 510. According to another embodiment of the present invention, the resulting elements of the execution are added to the existing values in the accumulator elements. That is, the accumulator "accumulates" (i.e., adds) the resulting elements onto its existing elements. The elements in the accumulator are then transformed into N-bit width in step 512. Finally, in step 514, the transformed elements are stored into memory. The process then terminates in step 516.

The SIMD vector instructions according to the present invention either write all 192 bits of the accumulator or all 64 bits of an FPR, or the condition codes. Results are not stored to multiple destinations, including the condition codes.

Integer vector operations that write to the FPRs clamp the values being written to the target's representable range. That is, the elements are saturated

for overflows and under flows. For overflows, the values are clamped to the largest representable value. For underflows, the values are clamped to the smallest representable value.

5           On the other hand, integer vector operations that write to an accumulator do not clamp their values before writing, but allow underflows and overflows to wrap around the accumulator's representable range. Hence, the significant bits that otherwise would be lost are stored into the extra bits provided in the accumulator. These extra bits in the accumulator thus ensure  
10   that unwanted overflows and underflows do not occur when writing to the accumulator or FPRs.

#### SELECTION OF VECTOR ELEMENTS

15           The preferred embodiment of the present invention utilizes an accumulator register and a set of vector registers in performing precision arithmetic operations. First, an exemplary vector register, vs, is used to hold a set of vector elements. A second exemplary vector register, vt, holds a selected set of vector elements for performing operations in conjunction with the elements in vector register, vs. The present invention allows an  
20   arithmetic instruction to select elements in vector register vt for operation with corresponding elements in other vector registers through the use of a well known crossbar method. A third exemplary vector register, vd, may be used to hold the results of operations on the elements of the registers described above. Although these registers (vs, vt, and vd) are used to  
25   associate vector registers with a set of vector elements, other vector registers are equally suitable for present invention.

360621-9406260

To perform arithmetic operations on desired elements of a vector, the present invention uses a well known crossbar method adapted to select an element of the vector register, vt, and replicate the element in all other element fields of the vector. That is, an element of vt is propagated to all other elements in the vector to be used with each of the elements of the other vector operand. Alternatively, all the elements of the vector, vt, may be selected without modification. Another selection method allows an instruction to specify as an element an immediate value in the instruction opcode vector field corresponding to vt and replicate the element for all other elements of vector vt. These elements thus selected are then passed onto the VU for arithmetic operation.

Figure 6 illustrates element select format for 4 16-bit elements in a 64-bit register. The exemplary vector register vt 600 is initially loaded with four elements: A, B, C, and D. The present invention allows an instruction to select or specify any one of the element formats as indicated by rows 602 through 610. For example, element B for vt 600 may be selected and replicated for all 4 elements as shown in row 604. On the other hand the vt 600 may be passed without any modification as in row 610.

Figure 7 illustrates element select format for 8 8-bit elements in a 64-bit register. The exemplary vector register vt 700 is initially loaded with eight elements: A, B, C, D, E, F, G, and H. The present invention allows an instruction to select or specify any one of the element formats as indicated by rows 702 through 718. For example, element G for vt 700 may be selected and replicated for all 8 elements as shown in row 714. On the other hand, the vt 700 may be passed without any modification as in row 718.



## ARITHMETIC INSTRUCTIONS

In accordance with the preferred embodiment of the present invention, arithmetic operations are performed on the corresponding elements of vector registers. The instruction is fetched from main memory and is loaded into a  
5 instruction register. It specifies the arithmetic operation to be performed.

In the following arithmetic instructions, the operands are values in integer vector format. The accumulator is in the corresponding accumulator  
10 vector format. The arithmetic operations are performed between elements of vectors occupying corresponding positions in the vector field in accordance with SIMD characteristics of the present invention. For example, an add operation between vs and vt actually describes eight parallel add operations between vs[0] and vt[0] to vs[7] and vt[7]. After an arithmetic operation has  
15 been performed but before the values are written into the accumulator, a wrapped arithmetic is performed such that overflows and underflows wrap around the Accumulator's representable range.

Accumulate Vector Add (ADDA.fmt). In the present invention  
20 ADDA.fmt instruction, the elements in vector registers vt and vs are added to those in the Accumulator. Specifically, the corresponding elements in vector registers vt and vs are added. Then, the elements of the sum are added to the corresponding elements in the accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then  
25 are written into the accumulator.

Figure 8 illustrates an exemplary ADDA.fmt arithmetic operation between elements of operand registers vs 800 and vt 802. Each of the registers 800, 802, and 804 contains 4 16-bit elements. Each letter in the elements (i.e., A, B, C, D, E, F, G, H, and I) stands for a binary number. FFFF is an

5 hexadecimal representation of 16-bit binary number, 1111 1111 1111 1111. The vs register 800 holds elements FFFF, A, B, and C. The selected elements of vt registers are FFFF, D, E, and F. The ADDA.fmt arithmetic instruction directs the VU to add corresponding elements: FFFF+FFFF (=1FFFD), A+D, B+E, and C+F. Each of these sums are then added to the corresponding existing  
10 elements (i.e., FFFF, G, H, and I) in the accumulator 804: FFFF+1FFFD, A+D+G, B+E+H, and C+F+I. The addition of the hexadecimal numbers, 1FFFD and FFFF, produces 2FFFC, an overflow condition for a general purpose 64-bit register. The accumulator's representable range is 48 bits in accordance with the present invention. Since this is more than enough bits  
15 to represent the number, the entire number 2FFFC is written into the accumulator. As a result, no bits have been lost in the addition and accumulation process.

Load Vector Add (ADDL.fmt). According to the ADDL.fmt instruction,  
20 the corresponding elements in vectors vt and vs are added and then stored into corresponding elements in the accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator 706.

25 Figure 9 illustrates an exemplary ADDL.fmt arithmetic operation between elements of operand registers vs 900 and vt 902. Each of the registers 900, 902, and 904 contains 4 16-bit elements. Each letter in the elements (i.e.,

A, B, C, D, E, and F) stands for a binary number. FFFF is an hexadecimal representation of 16-bit binary number, 1111 1111 1111 1111. The vs register 900 holds elements FFFF, A, B, and C. The selected elements of vt registers are FFFF, D, E, and F. The ADDA.fmt arithmetic instruction instructs the VU

5 to add corresponding elements: FFFF+FFFF , A+D, B+E, and C+F. The addition of hexadecimal numbers, FFFF and FFFF, produces 1FFFD, a technical overflow condition for a general purpose 64-bit register. The present invention wraps the number 1FFFD around the accumulator's representable range, which is 48 bits. Since this is more than enough bits to

10 represent the number, the entire number 1FFFD is written into the accumulator. As a result, no bits have been lost in the addition process.

Accumulate Vector Multiply (MULA.fmt). The MULA.fmt instruction multiplies the values in vectors vt and vs. Then the product is added to the

15 accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator.

Add Vector Multiply to Accumulator (MULL.fmt). The MULL.fmt instruction multiplies the values in vectors vt and vs. Then, the product is

20 written to the accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator.

Subtract Vector Multiply from Accumulator (MULS.fmt). In

25 MULS.fmt instruction, the values in vector vt are multiplied by the values in vector vs, and the product is subtracted from the accumulator. Any

overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator.

Load Negative Vector Multiply (MULSL.fmt). The MULSL.fmt instruction multiplies the values in vector vt with the values in vector vs. Then, the product is subtracted from the accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator.

Accumulate Vector Difference (SUBA.fmt). The present SUBA.fmt instruction computes the difference between vectors vt and vs. Then, it adds the difference to the value in the accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator.

Load Vector Difference (SUBL.fmt). According to SUBL.fmt instruction, the differences of vectors vt and vs are written into those in the accumulator. Any overflows or underflows in the elements wrap around the accumulator's representable range and then are written into the accumulator.

20

### ELEMENT TRANSFORMATION

After an arithmetic operation, the elements in the accumulator are transformed into the precision of the elements in the destination registers for further processing or for eventual storage into a memory unit. During the transformation process, the data in each accumulator element is packed to the precision of the destination operand. The present invention provides the following instruction method for such transformation.

Scale, Round and Clamp Accumulator (Rx.fmt). According to Rx.fmt instruction, the values in the accumulator are shifted right by the values specified in a vector field vt in the instruction opcode. This variable shift  
5 supports application or algorithm specific fixed point precision. The vt operands are values in integer vector format. The accumulator is in the corresponding accumulator vector format.

Then, each element in the accumulator is rounded according to a mode  
10 specified by the instruction. The preferred embodiment of the invention allows three rounding modes: 1) round toward zero, 2) round to nearest with exactly halfway rounding away from zero, and 3) round to nearest with exactly halfway rounding to even. These rounding modes minimize truncation errors during arithmetic process.

The elements are then clamped to either a signed or unsigned range of an exemplary destination vector register, vd. That is, the elements are saturated to the largest representable value for overflow and the smallest representable value for underflow. Hence, the clamping limits the resultant  
15 values to the minimum and maximum precision of the destination elements without overflow or underflow.

### SAVING ACCUMULATOR STATE

Since the vector accumulator is a special register, the present invention  
25 allows the contents of the accumulator to be saved in a general register. However, because the size of the elements of the accumulator is larger than the elements of general purpose registers, the transfer occurs in multiple

chunks of constituent elements. The following instructions allow storage of the accumulator state.

Read Accumulator (RAC.fmt). The RAC.fmt instruction reads a  
5 portion of the accumulator elements, preferably a third of the bits in  
elements, and saves the elements into a vector register. Specifically, this  
instruction method allows the least significant, middle significant, or most  
significant third of the bits of the accumulator elements to be assigned to a  
vector register such as vd. In this operation, the values extracted are not  
10 clamped. That is, the bits are simply copied into the elements of vector  
register, vd.

Write Accumulator High (WACH.fmt). The WACH.fmt instruction  
loads portions of the accumulator from a vector register. Specifically, this  
15 instruction method writes the most significant third of the bits of the  
accumulator elements from a vector register such as vs. The least significant  
two thirds of the bits of the accumulator are not affected by this operation.

Write Accumulator Low (WACL.fmt). According to WACL.fmt  
20 instruction, the present invention loads two thirds of the accumulator from  
two vector registers. Specifically, this instruction method writes the least  
significant two thirds of the bits of the accumulator elements. The remaining  
upper one third of the bits of the accumulator elements are written by the  
sign bits of the corresponding elements of a vector register such as vs,  
25 replicated by 16 or 8 times, depending on the data type format specified in the  
instruction.

A RACL/RACM/RACH instruction followed by WACL/WACH are used to save and restore the accumulator. This save/ restore function is format independent, either format can be used to save or restore accumulator values generated by either QH or OB operations. Data conversion need not occur. The mapping between element bits of the OB format accumulator and bits of the same accumulator interpreted in QH format is implementation specific, but consistent for each implementation.

The present invention, a method for providing extended precision in SIMD vector arithmetic operations, utilizes an accumulator register. While the present invention has been described in particular embodiments, it should be appreciated that the present invention should not be construed as being limited by such embodiments, but rather construed according to the claims below.

## CLAIMS

What is claimed is:

1. In a computer system including a processor which contains a first set of N-bit data elements loaded into a first register and a second set of N-bit data elements loaded into a second register, a method for providing extended precision in single instruction multiple data (SIMD) arithmetic operations, comprising the steps of:
  - fetching an arithmetic instruction from a memory unit;
  - decoding the arithmetic instruction and reading the first vector register and the second vector register;
  - executing the arithmetic instruction on corresponding N-bit data elements in the first register and second register to produce corresponding resulting elements;
  - writing the resulting elements into corresponding elements of an accumulator;
  - transforming the each resulting element in the accumulator into N-bits; and
  - writing the transformed elements of N-bit width into a third register.
2. The method as recited in Claim 1, wherein said decoding step further comprises the steps of:
  - selecting an element from the second register; and
  - copying the selected element into the other elements in the second register.



3. The method as recited in Claim 1, wherein said arithmetic instruction is an addition of corresponding vector elements in the first and second vector registers.

5 4. The method as recited in Claim 1, wherein said arithmetic instruction is a multiplication of corresponding vector elements in the first and second vector registers.

10 5. The method as recited in Claim 1, wherein said arithmetic instruction is a subtraction of second vector register elements from the first vector register elements.

15 6. The method as recited in Claim 1, wherein said accumulator is a register having an integer multiples of 64-bit width.

7. The method as recited in Claim 1, wherein said accumulator is a register of 192-bits.

20 8. The method as recited in Claim 1, wherein said transformation step further comprises the steps of:

scaling the resulting elements in the accumulator by shifting the values in the resulting elements;

rounding the scaled resulting elements in the accumulator; and  
clamping the rounded resulting elements.

25 9. The method as recited in Claim 1, wherein said third register writing step further comprises the steps of:

reading a portion of the accumulator elements; and  
writing the portion of the accumulator elements into the  
corresponding elements of said third register.

5           10.    The method as recited in Claim 9, wherein the portion is either  
the low third bits or the high third bits of the elements in the accumulator.

11.    The method as recited in Claim 1, wherein the values in the  
resulting elements are wrapped around the representable range of the  
10   accumulator elements.

12.    The method as recited in Claim 1, wherein the data elements are  
integers.

15           13.    The method as recited in Claim 1, wherein the first register, the  
second register, and the third registers are floating point registers.

14.    The method as recited in Claim 1, wherein the first register, the  
second register, and the third register are each 64-bit wide.  
20

15.    The method as recited in Claim 1, wherein N is 8.

16.    The method as recited in Claim 1, wherein N is 16.

25           17.    The method as recited in Claim 15, wherein the elements in the  
accumulator are each 24 bit wide.

18. The method as recited in Claim 16, wherein the elements in the accumulator are each 48 bit wide.

19. The method as recited in Claim 1, wherein said third register  
5 writing step further comprises the steps of:  
reading a portion of the accumulator elements; and  
writing the portion of the accumulator elements into the  
corresponding elements of said third register.

10 20. The method as recited in Claim 19, wherein the portion is  
chosen from the low third bits, the middle third bits, or the high third bits of  
the elements in the accumulator.

21. In a computer system including a processor which contains a  
15 first set of N-bit data elements loaded into a first register, a second set of N-bit  
data elements loaded into a second register, and an accumulator having a  
third set of data elements, a method for providing extended precision in  
single instruction multiple data (SIMD) arithmetic operations, comprising the  
steps of:

20 fetching an arithmetic instruction from a memory unit;  
decoding the arithmetic instruction and reading the first vector register  
and the second vector register;

executing the arithmetic instruction on corresponding data elements in  
the first and second vector registers to produce corresponding resulting  
25 elements;

adding the resulting elements to the corresponding elements in the  
accumulator;

writing the resulting elements into the accumulator;  
transforming the each resulting element in the accumulator into an N-bit width element; and  
writing the transformed elements of N-bit width into a third register.

5

22. The method as recited in Claim 21, wherein said decoding step further comprises the steps of:

selecting an element from the second register; and  
copying the selected element into the other elements in the second  
10 register.

23. The method as recited in Claim 21, wherein said arithmetic instruction is an addition of corresponding vector elements in the first and second vector registers.

15

24. The method as recited in Claim 21, wherein said arithmetic instruction is a multiplication of corresponding vector elements in the first and second vector registers.

20

25. The method as recited in Claim 21, wherein said arithmetic instruction is a subtraction of second vector register elements from the first vector register elements.

26. The method as recited in Claim 21, wherein said accumulator is  
25 a register having an integer multiples of 64-bit width.

27. The method as recited in Claim 21, wherein said accumulator is a register of 192-bits.

28. The method as recited in Claim 21, wherein said transformation  
5 step further comprises the steps of:

scaling the resulting elements in the accumulator by shifting the values in the resulting elements;

rounding the scaled resulting elements in the accumulator; and clamping the rounded resulting elements.

10

29. The method as recited in Claim 21, wherein said third register writing step further comprises the steps of:

reading a portion of the accumulator elements; and

15 writing the portion of the accumulator elements into the corresponding elements of said third register.

30. The method as recited in Claim 29, wherein the portion is either the low third bits or high third bits of the elements in the accumulator.

20 31. The method as recited in Claim 21, wherein the values in the resulting elements are wrapped around the representable range of the accumulator elements.

25 32. The method as recited in Claim 21, wherein the data elements are integers.

33. The method as recited in Claim 21, wherein the first register, the second register, and the third registers are floating point registers.

34. The method as recited in Claim 21, wherein the first register, the second register, and the third register are each 64-bit wide.

35. The method as recited in Claim 21, wherein N is 8.

36. The method as recited in Claim 21, wherein N is 16.

37. The method as recited in Claim 35, wherein the elements in the accumulator are each 24 bit wide.

38. The method as recited in Claim 36, wherein the elements in the accumulator are each 48 bit wide.

39. The method as recited in Claim 21, wherein said third register writing step further comprises the steps of:

reading a portion of the accumulator elements; and

writing the portion of the accumulator elements into the corresponding elements of said third register.

40. The method as recited in Claim 39, wherein the portion is chosen from the low third bits, the middle third bits, or the high third bits of the elements in the accumulator.

## ABSTRACT

The present invention provides extended precision in SIMD arithmetic operations in a processor having a register file and an accumulator. A first set of data elements and a second set of data elements are loaded into a first  
5 vector register and a second vector register, respectively. Each data element comprises N bits. Next, an arithmetic instruction is fetched from memory. The arithmetic instruction is decoded. Then, a first vector register and a second vector register are read from the register file. The present invention then executes the arithmetic instruction on corresponding data elements in  
10 the first and second vector registers. The result of the execution is then written into the accumulator. Then, each element in the accumulator is transformed into an N-bit width element and stored into the memory.

15

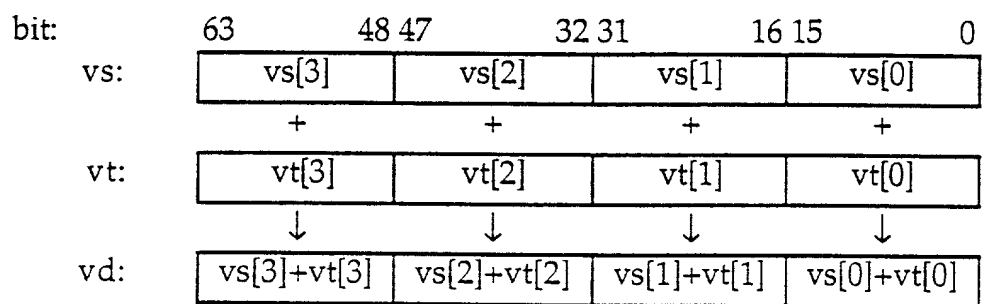


FIG. 1  
(Prior Art)



COMPUTER SYSTEM 212

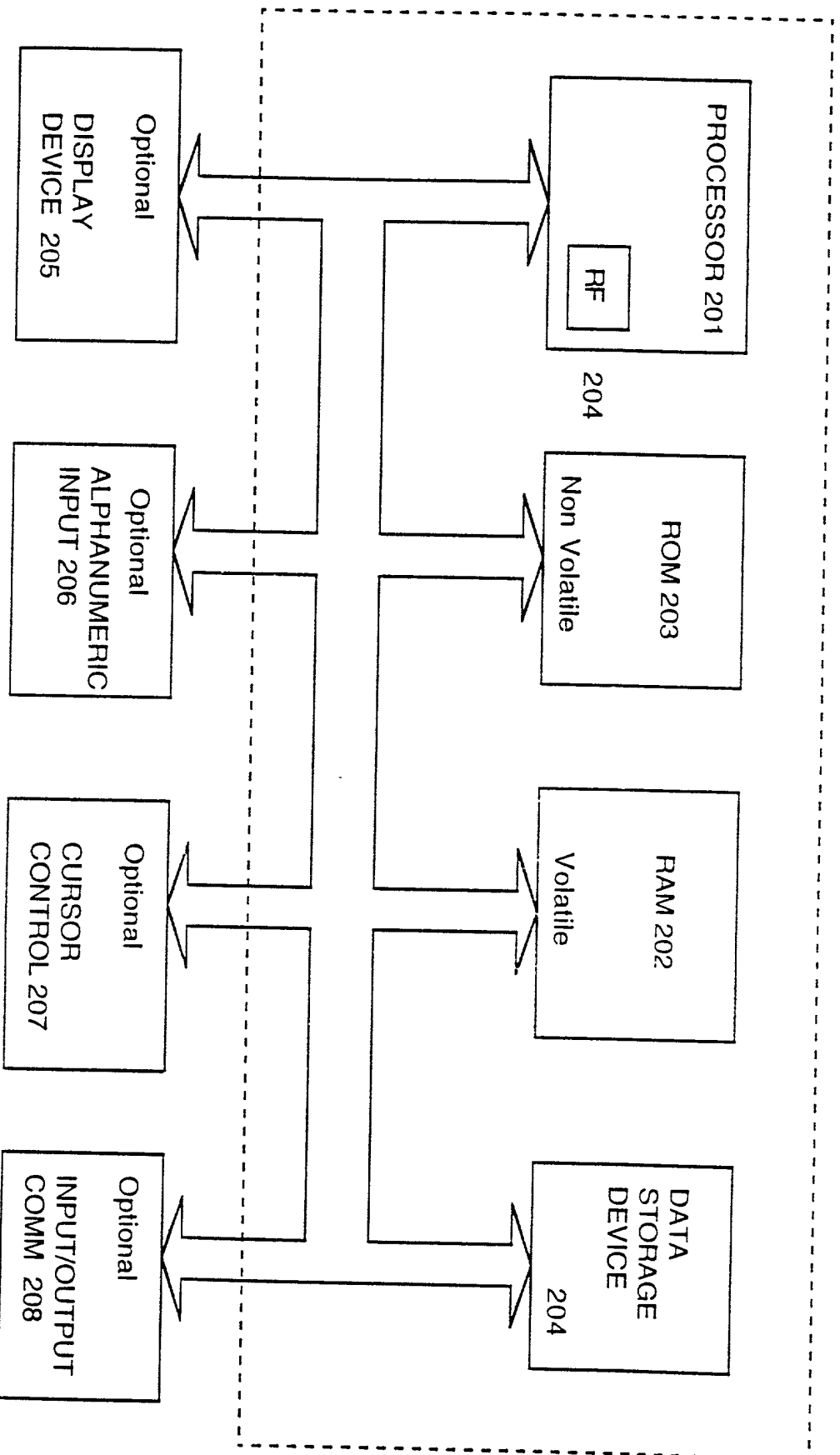
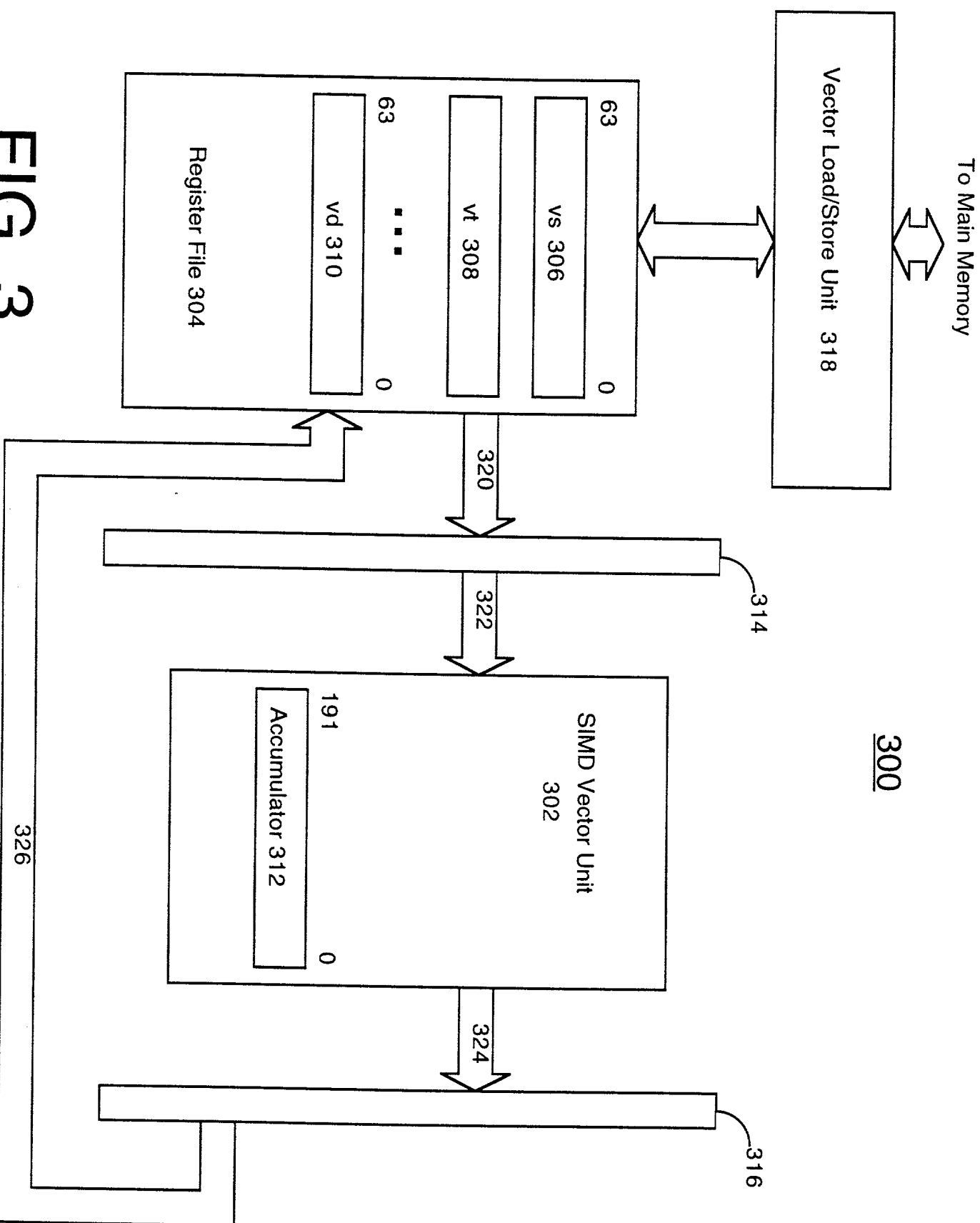


FIG. 2



300

FIG. 3

0923046-123098

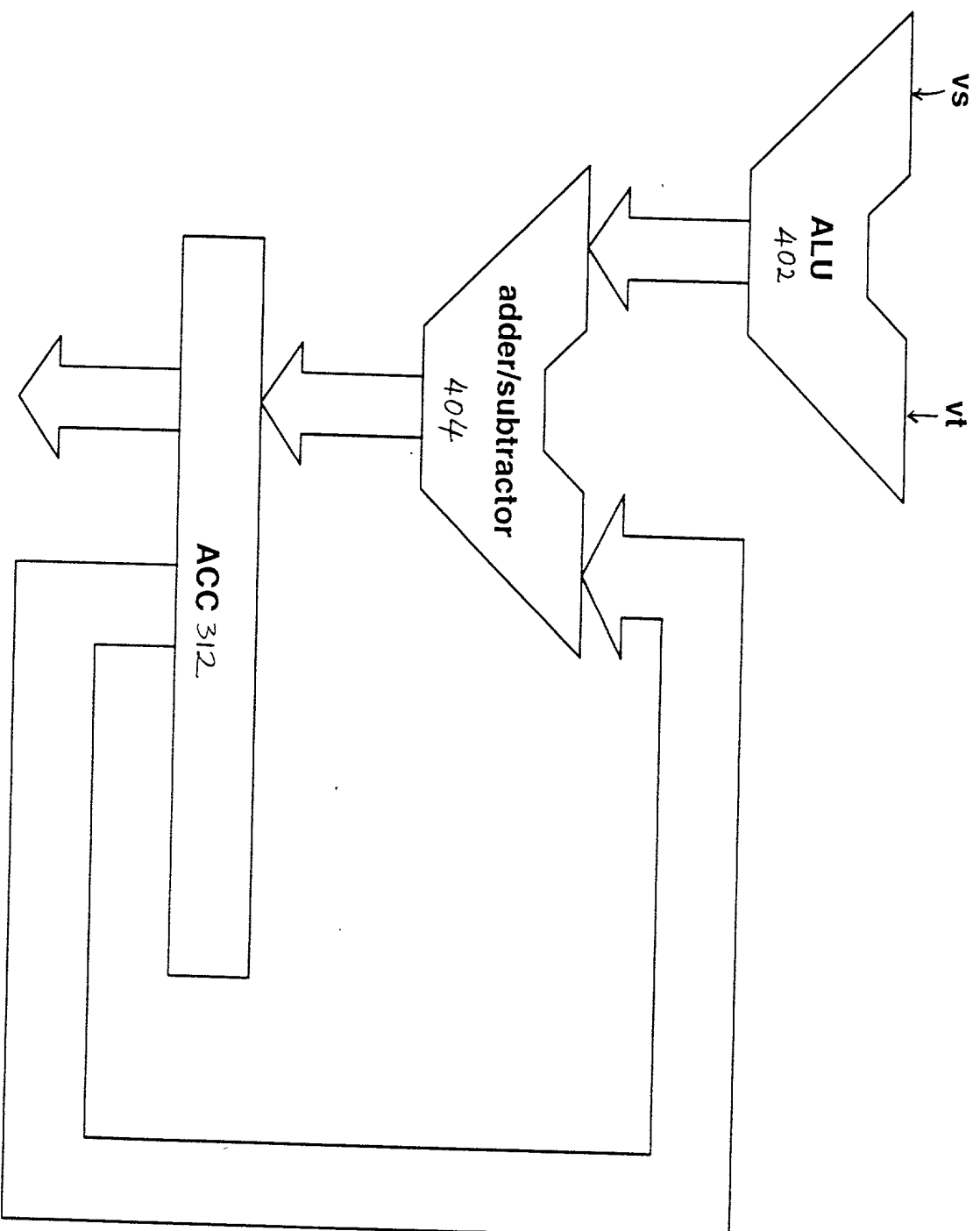
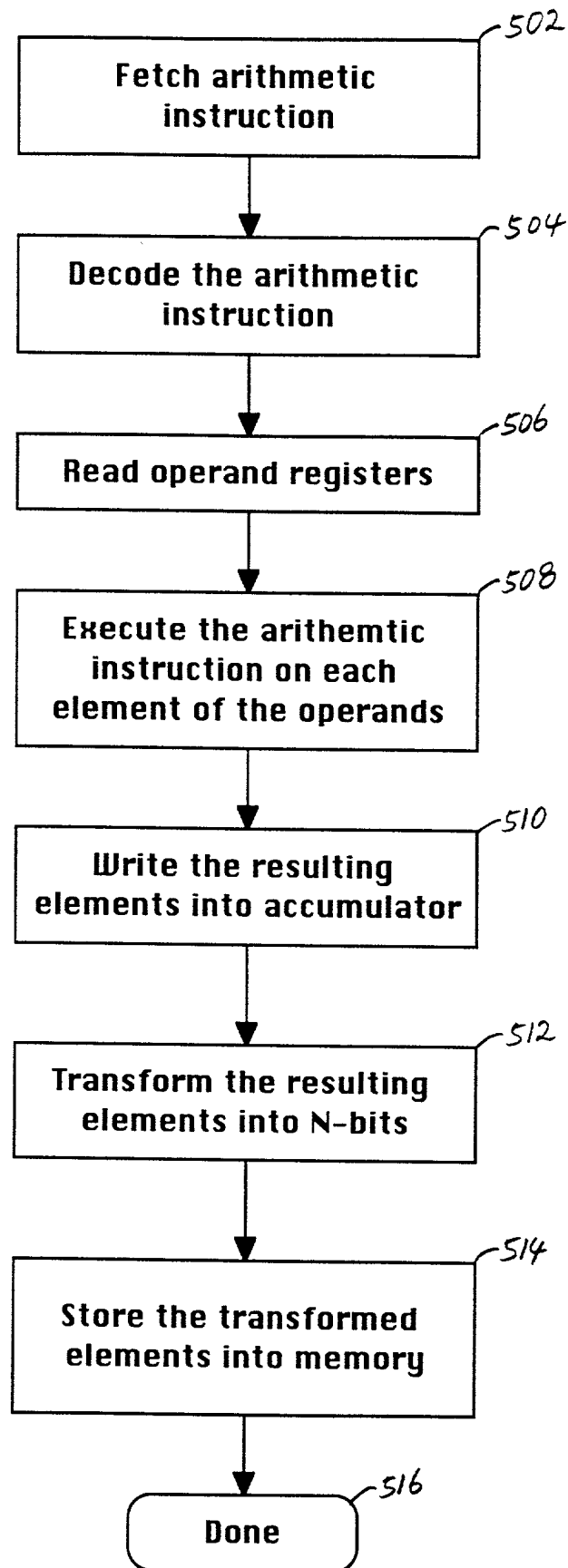


FIG. 4



**FIG. 5**

# QH DATA TYPE ELEMENT SELECT FORMAT

	63	48	47	32	31	16	15	0
vt:	D		C		B		A	

A	A	A	A	602
B	B	B	B	604
C	C	C	C	606
D	D	D	D	608
D	C	B	A	610

FIG. 6

OB DATA TYPE ELEMENT SELECT FORMAT

vt:

63	56	55	48	47	40	39	32	31	24	23	16	15	8	7	0
H	G	F	E	D	C	B	A								

700

A	A	A	A	A	A	A	A	A	702
B	B	B	B	B	B	B	B	B	704
C	C	C	C	C	C	C	C	C	706
D	D	D	D	D	D	D	D	D	708
E	E	E	E	E	E	E	E	E	710
F	F	F	F	F	F	F	F	F	712
G	G	G	G	G	G	G	G	G	714
H	H	H	H	H	H	H	H	H	716
H	G	F	E	D	C	B	A	A	718

FIG. 7

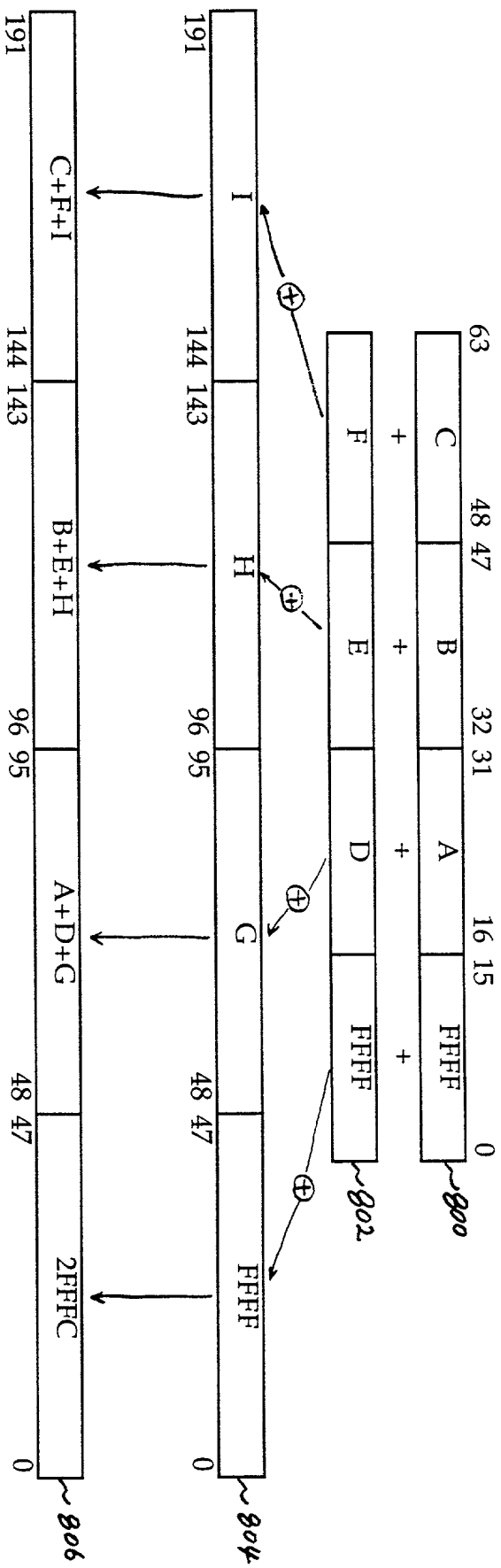


FIG. 8

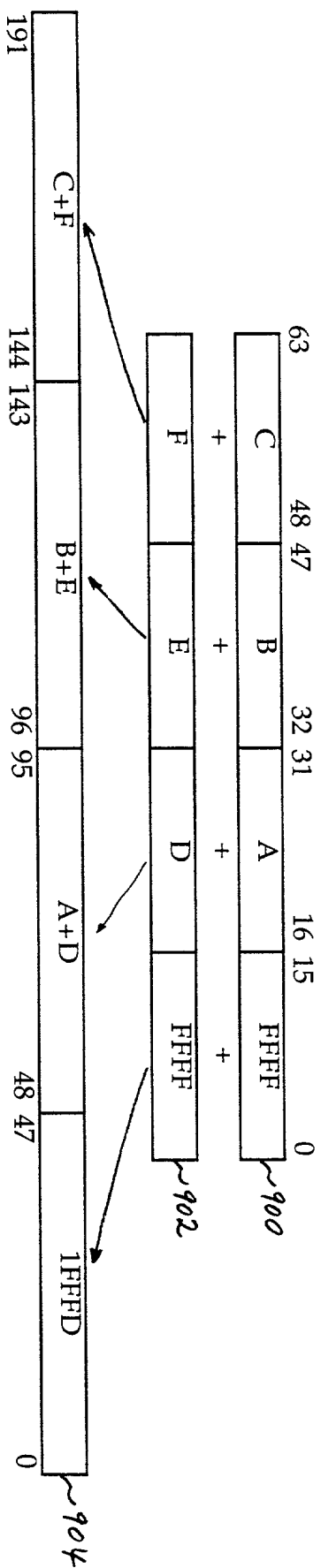


FIG. 9



## Declaration and Power of Attorney for a Patent Application

### Declaration

As below named inventor, I hereby declare that my residence post office address, and citizenship are as stated below my name. Further, I hereby declare that I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

A METHOD FOR PROVIDING EXTENDED PRECISION IN SIMD VECTOR ARITHMETIC OPERATIONS  
the specification of which:

..... is attached hereto, or  
☒ was filed on 10/9/97 as application serial no. 08/947,648 : and  
..... was amended on .....

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above; and

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

### Foreign Priority Claim

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Date Filed	Priority Claimed
.....	.....	.....	yes no
.....	.....	.....	yes no

### U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Serial Number	Filing Date	Status (patented/pending/abandoned)
.....	.....	.....
.....	.....	.....

## Power of Attorney

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent Trademark Office connected therewith.

James P. Hao	Registration No.: 36,398
Anthony C. Murabito	Registration No.: 35,295
John P. Wagner	Registration No.: 35,398
Glenn D. Barnes	Registration No.: P-42,293
Wilfred H. Lam	Registration No.: P-41,923
Steve Weiner	Registration No.: 38,330
Chris Byrne	Registration No.: 32,204
Irene Fernandez	Registration No.: 34,625
John Brigden	Registration No.: 40,530

Send Correspondence to:

**WAGNER, MURABITO & HAO**  
Two North Market Street, Third Floor  
San Jose, California 95113  
(408) 938-9060

## Signatures

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: Timothy van Hook

Inventor's Signature James Hao Date 5/11/98  
James P. Hao signing for Timothy J. van Hook under 37 C.F.R. § 1.47 (b)  
Residence Atherton, California Citizenship USA  
(City State)  
P.O. Address 224 Oakgrove Avenue, Atherton, California 94027

Full Name of Second/Joint Inventor: Peter Hsu

Inventor's Signature \_\_\_\_\_ Date \_\_\_\_\_  
Residence Fremont, California Citizenship \_\_\_\_\_  
(City State)  
P.O. Address 2853 Welk Common, Fremont, California 94555

Full Name of Third/Joint Inventor: William A. Huffman

Inventor's Signature \_\_\_\_\_ Date \_\_\_\_\_  
Residence Los Gatos, California Citizenship USA  
(City State)  
P.O. Address 16205 Roseleaf Lane, Los Gatos, California 95032

860831 94063260

Full Name of Fourth/Joint Inventor: Henry P. Moreton

Inventor's Signature \_\_\_\_\_ Date \_\_\_\_\_

Residence Woodside, California Citizenship USA  
(City State)

P.O. Address 140 Phillip Road, Woodside, California 94062-2625

Full Name of Fifth/Joint Inventor: Earl A. Killian

Inventor's Signature \_\_\_\_\_ Date \_\_\_\_\_

Residence Los Altos Hills, California Citizenship USA  
(City State)

P.O. Address 27961 Central Drive, Los Altos Hills, California 94022

260667-940622625